Améliorer la disponibilité de ses services

56

Posté par **Denis Dordoigne** le 23/07/14 à 13:11. Édité par **BAud**, **Nÿco**, **palm123**, **Tonton Th**, **Benoît Sibaud** et **Nils Ratusznik**. Modéré par **tankey**. **Licence CC by-sa**

Votre aventure d'hébergeur amateur prend de l'ampleur. Depuis quelques mois, vous avez réussi à gérer plusieurs services de façon transparente, mais maintenant que vous avez de plus en plus d'utilisateurs de vos services, vous vous rendez compte que votre unique serveur web est surchargé et que chaque maintenance provoque des coupures de service que ne comprennent pas vos visiteurs.

Afin de répondre à cette problématique, le plus simple est de multiplier les serveurs : la charge sera répartie entre les différents serveurs et vous pourrez couper un serveur pour une maintenance, sans couper le service associé.

Sommaire

- Étape 1 : digérer quelques concepts
  - Répartition de charge
  - Adresse IP virtuelle
  - Test de vie
  - Ordonnanceur de répartition
  - Un peu de routage
    - Méthode 1 : tout passe par le répartiteur de charge
    - Méthode 2 : faisons travailler le serveur
- Étape 2 : multiplier les serveurs
  - Mon service est-il multipliable ?
  - Mes ressources sont-elles accessibles de partout ?
  - Comment gérer mon routage ?
    - Définition d'un routage statique
    - Gérer le routage direct
- Étape 3 : mon premier répartiteur de charge
  - installation
    - Machines
    - Paquets
    - Paramètres système
    - Configuration de base
    - Configuration de l'instance
  - Ma première adresse IP virtuelle
    - Rappel des pré-requis pour le routage direct
    - Déclaration dans keepalived.conf
  - On s'en fait une deuxième ?
    - Choix de l'adresse IP
    - Ajout de quelques options
    - Choix du test de vie
      - GET d'une URL
      - test personnalisé
- Étape 4 : Exploitons tout ça
  - Extraire des statistiques
  - Manipuler vos adresses virtuelles dynamiquement

Étape 1 : digérer quelques concepts

Afin de simplifier les explications et de coller à ce qui est probablement votre cas d'usage, nous considérerons dans ce tutoriel que nous travaillons dans une infrastructure totalement IPv4 [W].

Répartition de charge

La répartition de charge [W] « *est un ensemble de techniques permettant de distribuer une charge de travail entre différents ordinateurs d'un groupe. Ces techniques permettent à la fois de répondre à une charge trop importante d'un service en la répartissant sur plusieurs serveurs, et de réduire l'indisponibilité potentielle de ce service que pourrait provoquer la panne logicielle ou matérielle d'un unique serveur* »[1] . On distingue deux grands types de répartition de charge :

- la répartition parallèle (« actif / actif ») : plusieurs serveurs offrent de façon simultanée le service, le répartiteur de charge peut envoyer une requête indifféremment à chaque serveur ;
- la répartition séquentielle (« actif / passif ») : plusieurs serveurs sont capables de rendre le service, mais le répartiteur de charge n'envoie des requêtes qu'à un seul d'entre eux ; l'envoi de requête à

un autre serveur ne sera fait que si le serveur nominal n'est plus en mesure de prendre en compte les requêtes.

Le mode séquentiel ne permet pas de répartir la charge de travail à proprement parler puisque seul un serveur rend le service à la fois, en revanche cela répond bien au besoin de ne pas interrompre le service en cas de coupure d'un serveur.

Adresse IP virtuelle

Une adresse IP virtuelle (parfois appelée « vIP » ou « serveur virtuel », même si l'usage de ce dernier terme est tombé en désuétude suite à l'arrivée des « machines virtuelles » et le risque de confusion associé) est l'adresse IP d'un service faisant l'objet d'une répartition de charge : l'adresse est dite virtuelle parce qu'elle n'est portée par aucun serveur à proprement parler, mais par un groupe de serveurs, défini dans la configuration du répartiteur de charge. L'adresse IP virtuelle peut être utilisée comme n'importe quelle adresse IP, on doit notamment en autoriser l'accès depuis le pare-feu comme on le ferait pour l'adresse IP d'un serveur.

Test de vie

Puisqu'on demande au répartiteur de charge de n'envoyer les requêtes qu'aux serveurs en état de les traiter, il faut lui donner les moyens de définir quels serveurs sont hors-service. Pour cela, on va configurer un ou plusieurs tests de vie par service, par exemple :

- un ping du serveur cible (test rarement pertinent, mais dans le cadre d'un réseau local non filtré et non routé, on peut considérer que si un serveur ne répond pas au ping c'est qu'il n'est plus en état de rendre le service)
- une connexion TCP sur un port (si le serveur ne permet pas d'ouvrir une connexion sur le port 443, il ne rend à priori pas le service HTTPS)
- un test applicatif (pour un serveur HTTP on peut vérifier qu'un « GET / » ne renvoie pas une erreur 500)
- un test de service (pour un serveur HTTP hébergeant un wiki on peut aller jusqu'à tester qu'une modification de page réussit).

Les tests de vie sont associés à une fréquence d'exécution, qui définira la durée maximale pendant laquelle on accepte qu'un serveur HS continue de recevoir des requêtes (par exemple, si on veut que le serveur ne reçoive plus de requêtes moins de 2 secondes après être tombé, il faut mettre en place un test de vie toutes les secondes), il faut donc veiller :

- à avoir un test de vie plus rapide que votre fréquence d'exécution (si vous lancez un test qui prend 5 secondes chaque seconde, ceux-ci vont s'empiler sur les serveurs) ;
- à ne pas avoir des tests de vie qui deviennent une cause de surcharge des serveurs rendant le service (pour reprendre l'exemple du wiki, si vous avez 30 éditions par jour habituellement en faisant une édition par test de vie vous allez subitement en avoir des milliers).

Ordonnanceur de répartition

Dans le cadre d'une répartition parallèle, chaque requête vers une adresse IP virtuelle est envoyée à un ordonnanceur qui se charge de définir par quel serveur la requête doit être traitée, parmi tous les serveurs détectés comme vivants. Il y a 3 grandes familles d'ordonnanceurs :

- les ordonnanceurs impartiaux : si on a 1000 requêtes réparties entre 4 serveurs, chaque serveur en traitera 250 ; pour cela l'algorithme généralement utilisé est le round-robin[W] (les serveurs reçoivent une requête chacun leur tour), mais certains répartiteurs de charge proposent également des ordonnanceurs basés sur un algorithme aléatoire[W]
- les ordonnanceurs compensateurs : la requête est envoyée au serveur qui la traitera le plus vite (l'algorithme généralement utilisé est d'envoyer au serveur qui a le moins de connexions actives), l'ordonnanceur se charge donc de compenser l'éventuelle lenteur d'un serveur en lui envoyant moins de requêtes ; attention : même si cela n'est pas intuitif, un serveur défaillant traite souvent les requêtes plus rapidement qu'un serveur fonctionnel (un accès refusé à une base de données peut prendre quelques millisecondes quand le traitement d'une requête peut prendre plusieurs secondes), donc ce type d'ordonnanceur favorisera les serveurs défaillants si votre test de vie n'est pas suffisamment bien conçu pour que ceux-ci ne soient plus considérés comme vivants
- les ordonnanceurs déterministes : une fonction de hachage[W] appliquée à la requête reçue permet de définir le serveur qui traitera la requête ; il y a deux principaux types de déterminisme :
  - déterminisme réseau[W] : une même adresse IP source (ou un même couple adresse/port) enverra toujours au même serveur ; à noter que si vous avez de nombreux utilisateurs derrière un même proxy la répartition ne sera pas optimale ;
  - déterminisme applicatif[W] : la même demande (par exemple "GET /login.php") enverra toujours au même serveur (en général la requête est analysée au niveau de la couche application[W], l'analyse du paquet TCP n'étant pas suffisante pour calculer un hash pertinent).

Les ordonnanceurs acceptent parfois des options :

- gestion des poids : on peut donner des poids différents aux serveurs pointés par une adresse virtuelle afin que ceux-ci soient privilégiés par l'algorithme de répartition (par exemple un ordonnanceur impartial enverra deux fois plus de connexions à un serveur de poids 10 qu'à un

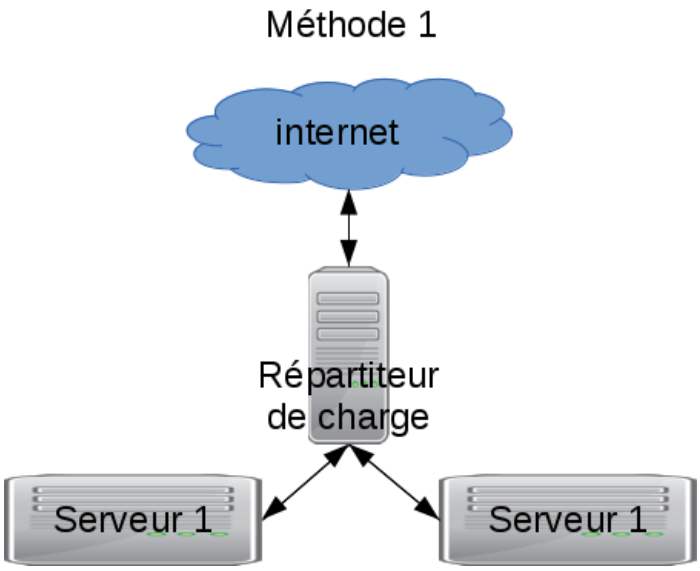
serveur de poids 5).

- persistance de session : l'ordonnanceur n'est appelé que pour la première connexion d'un client, puis le répartiteur de charge conserve dans une table de sessions le serveur cible associé à ce client : tant que ce serveur sera vu vivant, toutes les requêtes du client lui seront envoyées.

Un peu de routage

Vous allez donc avoir des connexions qui vont arriver depuis vos répartiteurs de charge vers vos serveurs ; maintenant, il faut se poser une question : comment répondre au client qui a fait la requête ? Il y a deux écoles, chacune ayant ses avantages et inconvénients.

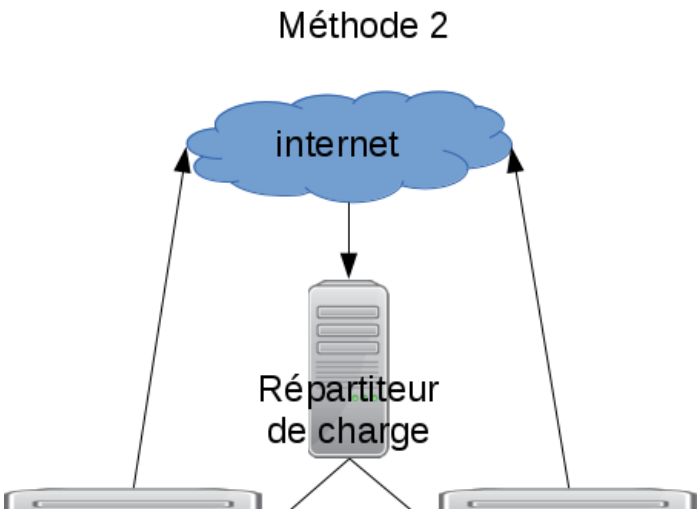
Méthode 1 : tout passe par le répartiteur de charge



Les connexions arrivent au répartiteur de charge ? Qu'elles y retournent ! Cette méthode qui est la plus utilisée consiste à répondre aux requêtes envoyées par le répartiteur de charge au répartiteur de charge lui-même, celui-ci s'occupant de les renvoyer sur Internet. Il y a deux façons de procéder :

- le NAT source : le repartiteur de charge se présente au serveur avec sa propre adresse IP, la réponse est faite naturellement à cette adresse
  - avantages : cela fonctionne avec à peu près tous les services imaginables sans avoir à modifier le serveur cible, si on n'a pas que du logiciel libre côté serveur cela simplifiera grandement les choses ;
  - inconvénients : comme dans le cas d'un proxy, le serveur ne verra pas l'adresse IP d'origine, cela complique la gestion des traces que l'on doit conserver dans le cadre de la loi pour la confiance dans l'économie numérique, le diagnostic des problématiques rencontrées par certains utilisateurs et la mise en place de contrôles d'accès.
- le routage statique : le répartiteur de charge envoie les connexions telles quelles au serveur, mais celui-ci dispose d'un routage statique pour renvoyer toutes les requêtes provenant d'Internet au répartiteur de charge
  - avantages : on n'a pas les inconvénients du NAT ;
  - inconvénients : il faut maintenir une table de routage pour chaque serveur en y listant l'ensemble des réseaux et services auxquels on est susceptible de devoir accéder sans passer par le répartiteur de charge.

Méthode 2 : faisons travailler le serveur



Cette méthode consiste à déléguer au serveur la réponse aux clients, sans repasser par le répartiteur de charge :

- avantage : le répartiteur de charge se comporte comme un simple routeur, il consomme donc peu de ressources système, une machine virtuelle minuscule est suffisante pour rendre ce service ;
- inconvénient : cela nécessite de bidouiller les serveurs pour que ceux-ci acceptent de gérer des communications réseau peu orthodoxes et, en général, et dans ce cas hors système Linux cela s'avère complexe à mettre en œuvre.

Il y a deux façons de gérer ces connexions :

- le routage direct : on fait croire à chaque serveur qu'il est porteur de l'adresse IP virtuelle afin qu'il traite les connexions concernant cette adresse IP comme une connexion à une adresse IP locale ;
- le tunnel IP-IP : le répartiteur de charge envoie la connexion dans un tunnel et le serveur traite les connexions venant de son interface tunnel comme une connexion à une adresse IP locale (on préfère cette méthode au routage direct uniquement quand le répartiteur de charge n'est pas dans le même réseau que le serveur).

Étape 2 : multiplier les serveurs

Maintenant que vous savez que vous pouvez repartir la charge entre plusieurs serveurs, vous allez pouvoir commencer à multiplier ceux-ci : attention cependant à vous poser les bonnes questions.

Mon service est-il multipliable ?

Certains services nécessitant une ressource locale ne peuvent pas faire l'objet d'une répartition de charge. Par exemple, une base SQLite ne garantit sa cohérence que si elle est capable de poser un verrou sur un fichier : un verrou de fichier étant local à un serveur, il n'est pas possible de partager une telle base de données entre différents serveurs. Dans ce type de cas, un répartiteur de charge séquentiel peut devenir intéressant : on peut installer plusieurs serveurs mais demander au répartiteur de charge de n'en adresser qu'un seul à la fois, ainsi toutes les requêtes accéderont à la même ressource locale.

Mes ressources sont-elles accessibles de partout ?

Votre service utilise probablement des fichiers locaux et/ou des informations en mémoire pour fonctionner, il convient donc de s'assurer que celles-ci sont accessibles par tous les serveurs rendant le service. Il faut surtout se poser la question des informations de session que peut porter le service : celles-ci doivent être dans un espace partagé (il est commun de stocker des sessions php dans un montage NFS par exemple) ou dans un outil qui sait gérer la replication (une base de données en réseau par exemple). Si vous ne pouvez pas partager ou répliquer les informations de session entre vos différents serveurs, il conviendra de veiller à ce qu'un client ne change jamais de serveur pendant sa session, soit en utilisant la fonctionnalité de persistance de session de votre répartiteur de charge, soit en utilisant un ordonnanceur déterministe.

Comment gérer mon routage ?

On peut se contenter de faire du NAT et ne pas se poser la question. C'est même la solution préconisée par de nombreux outils de répartition de charge. Cependant, si le NAT ne répond pas à votre besoin pour une des raisons indiquées précédemment (obligation légale, contrôle d'accès, besoin d'investigation) ou tout simplement parce que votre applicatif ou votre protocole ne le gère pas, la mise en place d'une infrastructure répartie peut affecter la configuration de votre serveur.

Définition d'un routage statique

Le plus simple lorsqu'on fait le choix d'un routage statique est de configurer le répartiteur de charge comme passerelle par défaut de votre serveur. Cependant, si votre serveur ne fait pas que répondre à des requêtes en utilisant des ressources locales (par exemple s'il s'agit d'un serveur mail, il doit aussi communiquer avec le reste du monde pour envoyer des mails), il va falloir gérer un routage différent pour ces autres besoins : si votre applicatif le permet vous pouvez envoyer ces connexions à une interface réseau spécifique qui ne passera pas par la passerelle par défaut, sinon il faudra envisager l'usage d'un système de routage intelligent.

Gérer le routage direct

Si vous avez une plate-forme 100% Linux, n'hésitez pas à faire ce choix, il faudra juste autoriser le trafic d'ARP en ajoutant cela dans votre sysctl.conf :

```
net.ipv4.conf.all.arp_ignore=1
net.ipv4.conf.all.arp_announce=2
net.ipv4.conf.eth0.arp_ignore=1
net.ipv4.conf.eth0.arp_announce=2
```

(il faut le faire pour all et pour l'interface avec laquelle vous communiquez avec le répartiteur de charge) ; si vous n'avez pas prévu de redémarrer votre serveur, vous pouvez forcer la prise en compte de vos modifications :

```
# sysctl -p
```

Ensuite, il convient de faire comprendre au serveur qu'il gère l'adresse IP virtuelle. Le plus propre pour faire cela est de déclarer celle-ci comme un alias de l'interface de loopback :

```
# ifconfig lo:9
lo:4      Link encap:Boucle locale
          inet adr:192.168.10.9  Masque:255.255.255.255
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
```

Étape 3 : mon premier répartiteur de charge

Linux Virtual Server[W], abrégé en LVS, est un logiciel répartiteur de charge pour GNU/Linux. Il peut se configurer simplement en ligne de commande, mais afin de gérer simplement la configuration on utilise en général un logiciel spécialisé, dans ce tutoriel ce sera *keepalived*.

installation

Machines

Pour installer ce service, une simple machine virtuelle avec quelques centaines de Mo d'espace disque et quelques dizaines de Mo de mémoire suffira. Et puis comme on veut gérer la redondance en cas de panne, on va même en installer deux !

Paquets

Sur une Debian fraîchement installée avec le système de base, installer le paquet **keepalived** avec toutes ses dépendances mais sans les paquets recommandés :

```
# apt-get install keepalived
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  ipvsadm libnl1
Paquets suggérés :
  heartbeat ldirectord
Les NOUVEAUX paquets suivants seront installés :
  ipvsadm keepalived libnl1
0 mis à jour, 3 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 331 ko dans les archives.
Après cette opération, 995 ko d'espace disque supplémentaires seront utilisés.
```

Bien entendu ça fonctionne aussi bien avec d'autres distributions…

Paramètres système

Ajouter le paramètre suivants dans **/etc/sysctl.conf*** :

```
# Parametres pour le LVS
net.ipv4.ip_forward=1
```

Charger la configuration :

```
# sysctl -p
net.ipv4.ip_forward = 1
```

Configuration de base

Source : http://www.keepalived.org/documentation.html

Debian ne génère aucune configuration à l'installation, il faut donc créer le fichier **/etc/keepalived/keepalived.conf** après l'installation. Pour commencer, il faut y mettre la section globaldefs qui permet de définir la configuration de base :

```
global_defs {
  notification_email {
    georgette@example.com
  }
  notification_email_from lvs@example.com
  smtp_server relayhost.example.com
  smtp_connect_timeout 30
  router_id LVS
}

vrrp_sync_group VG1 {
 group {
 linuxfr
 }
}
```

Voici l'explication des paramètres :

- **notification_email** : liste des adresses (séparées par des sauts de ligne) notifiées en cas de changement d'état d'une adresse IP virtuelle (enverra par exemple un e-mail lorsqu'un serveur ne répond plus) ; ne pas mettre cette ligne si on ne désire pas être notifié

- **notification_email_from**, **smtp_server**, **smtp_connect_timeout** : paramètres d'expédition des mails
- **router_id** : le petit nom donné au service LVS, comme on n'en a qu'un sur la plateforme, on va faire simple en mettant *LVS*, mais on peut faire plus intelligent
- **group** : liste des instances déclarées (voir ci-dessous)

Configuration de l'instance

Toujours sans keepalived.conf, on peut créer plusieurs instances ayant chacune leur configuration (par exemple "prod" et "dev"), pour commencer on ne va en créer qu'une, nommée linuxfr :

```
vrrp_instance linuxfr {
 state MASTER
 interface eth0
 smtp_alert
 virtual_router_id 51
 authentication {
 auth_type PASS
 auth_pass 1111
 }
 virtual_ipaddress {
 192.168.2.9
 }
}
```

Explication des paramètres :

- **vrrp_instance** : début du bloc de paramètres de l'instance, doit être suivi du nom de l'instance (ici *linuxfr*)
- **state** : il s'agit de la seule différence entre la configuration du LVS primaire et du LVS secondaire : l'un d'entre eux doit être "MASTER", l'autre "SLAVE" (c'est pour cela qu'on peut se permettre de metre en place deux serveurs dès le début, la mise en place du second tient à un copier/coller suivi de cette seule modification)
- **interface** : nom de l'interface surveillée par le service
- **smtp_alert** : à positionner ou non selon que l'on souhaite avoir des notification par courriel des défaillances
- **virtual_router_id** : on peut mettre n'importe quel nombre entre 0 et 255, il faut juste qu'il soit différent entre les différentes instances
- **authentication**, **auth_type**, **auth_pass** : identifiants utilisés par les serveurs LVS pour communiquer entre eux, notez cependant que ces informations circulent en clair dans le réseau
- **virtual_ipaddress** : liste des adresses virtuelles portées par le LVS (une par ligne, 20 maximum) ; c'est la partie que vous oublierez systématiquement de remplir en ajoutant de nouvelles adresses IP virtuelles, et vous perdrez 5 minutes à chercher pourquoi ça ne marche pas

Ma première adresse IP virtuelle

Dans cet exemple, nous déclarons une adresse virtuelle suivante 192.168.10.9 qui renvoie le port 80 vers le port 80 des serveurs 192.168.10.98 et 192.168.10.85.

Rappel des pré-requis pour le routage direct

- vos répartiteurs de charge doivent être dans le même réseau que vos serveurs (sinon configurez votre LVS pour utiliser des tunnels)
- vos serveurs doivent accepter le trafic ARP entre leurs interfaces (cf. paramètres systcl plus haut, si vous n'avez pas la main sur vos serveurs, configurez votre LVS pour faire du NAT source)
- chaque serveur doit croire qu'il porte l'adresse IP du service (ici on déclarera un alias de l'interface de loopback avec l'adresse IP 192.168.10.9)
- l'adresse doit être connue du bloc virtual_ipaddress de votre instance LVS (certes on vous l'a déjà précisé dans le paragraphe précédent, mais on sait que vous allez l'oublier)

Déclaration dans keepalived.conf

```
virtual_server 192.168.10.9 80 {
 delay_loop 6
 lb_algo rr
 lb_kind DR
 protocol TCP

 real_server 192.168.10.98 80 {
 weight 1
 TCP_CHECK {
  connect_port 80
  connect_timeout 3
```

```
  }
 }

 real_server 192.168.10.85 80 {
 weight 1
 TCP_CHECK {
  connect_port 80
  connect_timeout 3
  }
 }
}

}
```

- **virtual_server** : doit être suivi de l'adresse IP virtuelle puis du port
- **delay_loop** : délai entre deux tests de vie (n'oubliez pas que vous avez deux serveurs LVS, donc vos serveurs se prendront deux fois la charge correspondante)
- **lb_algo** : l'algorithme utilisé par l'ordonnanceur ; les plus utilisés sont rr (round-robin) et lc (moins de connexions actives) avec leurs équivalents wrr et wlc prenant en compte les poids ; la liste complète des algorithmes est disponible dans http://www.linuxvirtualserver.org/docs/scheduling.html
- **lb_kind** : méthode d'accès aux serveurs, pour du routage direct on indique "'DR'"
- **real_server** : doit être suivi de l'adresse IP d'un serveur et du port du service. Il faut autant de blocs real_server qu'il y a de serveurs derrière l'adresse virtuelle
- **weight** : poids, notamment utilisé pour les algorithmes wlc et wrr ; par défaut, le poids est 1
- **TCP_CHECK** : test de vie de type ouverture de connexion TCP ; dans cette exemple si une connexion au port 80 prend plus de 3 secondes, le serveur n'est plus considéré comme vivant

On s'en fait une deuxième ?

Vous avez probablement plus d'un service à répartir, donc il faudra créer une adresse IP virtuelle par service.

Choix de l'adresse IP

Pour votre deuxième service vous pouvez soit attribuer une nouvelle adresse IP, soit réutiliser celle d'une adresse de service existante, à condition évidemment que ce soit sur un port différent (et en plus comme ça elle est déjà dans le bloc virtual_servers, vous ne l'oublierez pas pour une fois).

Ajout de quelques options

```
persistence_timeout 60

virtualhost supervision.fr.local

quorum 30
hysteresis 2
quorum_up "/usr/local/bin/notify.pl qourum up"
quorum_down "/usr/local/bin/start_spare_vm.pl"

sorry_server 192.168.10.55 80
```

- persistence_timeout : mettre un nombre de secondes si on veut activer la persistance de session ; pendant ce nombre de secondes, une même adresse IP source sera systématiquement envoyée au même serveur sans que l'ordonnanceur ne soit sollicité
- quorum : poids total des serveurs actifs nécessaire pour considérer l'adresse virtuelle comme pleinement opérationnelle
- quorum_down : commande à lancer quand le quorum n'est plus atteint, en général on met une commande qui envoie une alarme par sms ou dans l'outil de supervision, mais selon votre architecture vous pouvez aussi envisager de démarrer automatiquement des serveurs supplémentaires, d'activer une version allégée de vos services, etc.
- quorum_up : commande à lancer une fois que le quorum est de nouveau atteint
- hystérésis : différence de poids minimum entre deux appels de commande quorum_up/quorum_down ; par exemple dans notre cas si un quorum_down a été détecté à 29, le quorum_up ne sera pas appelé lors du passage à 30 mais seulement lors du passage à 31 ; cette fonctionnalité permet d'éviter d'appeler les commandes trop souvent lorsqu'on est proche des limites, c'est surtout utile si la commande appelée est particulièrement lourde
- sorry_server : serveur auquel seront envoyées les requêtes si aucun des serveurs pointés par l'adresse virtuelle ne répond ; pour un service web ça pourrait être un mini serveut hébergeant une simple page html d'excuses
- virtualhost : pour un service web, nom de domaine vers lequel seront envoyés les tests de vie HTTP ou HTTPS (cf. paragraphe suivant)

Choix du test de vie

Pour notre première adresse IP virtuelle nous avons choisi un test de vie TCP, mais keepalived permet d'autres tests de vie.

GET d'une URL

```
HTTP_GET
{
  url
  {
    path /test_vie.php
    digest 5f1a4b7e269b7f5ddf6bbce06856c1e8
    status_code 200
  }
  connect_port 80
  connect_timeout 3
}
```

Si la page test_vie.php n'est pas dans le virtual host par défaut de votre serveur web, il faudra préciser le paramètre *virtualhost* dans la configuration de l'adresse IP virtuelle. Le paramètre *digest* correspond au hash MD5 de la réponse du serveur, on peut le récupérer avec la commande suivante :

```
# genhash -s 192.168.10.85 -p 80 -u /test_vie.php
MD5SUM = 5f1a4b7e269b7f5ddf6bbce06856c1e8
```

Veuillez noter que :

- on peut ne mettre qu'une seule information entre *digest* et *status_code* (code retour HTTP, a priori ce sera 200)
- on peut déclarer autant de blocs url{} que l'on veut dans un test, le serveur ne sera plus vu vivant si un seul d'entre eux échoue
- si on veut faire un test en HTTPS, il faut nommer le bloc SSL_GET au lieu de HTTP_GET (et pour récupérer le digest ajouter l'option « -S » à la commande genhash)
- le digest dépend du contenu de la page, n'ayez pas de contenu dynamique dedans ! Afficher l'heure ou la durée d'affichage par exemple ferait tomber systématiquement le test en erreur ; par contre c'est utile pour valider que tout va bien, par exemple on peut faire une page qui affiche « OK » quand elle arrive à accéder à la base de données, et « KO » sinon : le digest n'étant retrouvé que lorsque la page affiche OK, le répartiteur de charge n'enverra pas de trafic aux serveurs incapables d'accéder à la base de données

test personnalisé

Il est possible d'écrire un script qui sera utilisé pour les checks, par exemple pour tester qu'un serveur LDAP est opérationnel, on fera un script qui fait une requête LDAP :

```
MISC_CHECK
{
    misc_path "/usr/local/bin/test_ldap.pl 192.168.10.72"
    misc_timeout 15
    # misc_dynamic
}
```

Le script doit simplement renvoyer 0 si le serveur est vivant, et une autre valeur si ce n'est pas le cas. Ici on n'a pas opté pour l'option misc_dynamic (elle est commentée), mais on peut l'activer si on utilise les algorithmes wrr ou wlc, dans ce cas le code retour du script sera interprété ainsi :

- 0: le serveur est vivant, son poids doit rester celui configuré dans keepalived.conf
- 1 : le serveur n'est pas vivant, plus aucune requête ne doit lui être envoyé
- de 2 à 255 : le serveur est vivant, mais son poids doit être changé par la valeur renvoyée moins deux (par exemple si le script a un code retour de 10 le nouveau poids du serveur sera 8)

Étape 4 : Exploitons tout ça

Votre service est configuré, il n'y a plus qu'à le lancer !

```
# service keepalived start
```

Keepalived permet seulement de gérer une configuration pour LVS, sans donner d'outils d'exploitation supplémentaires. Pour ensuite suivre la vie de votre service LVS, il faut utiliser la commande **ipvsadm**.

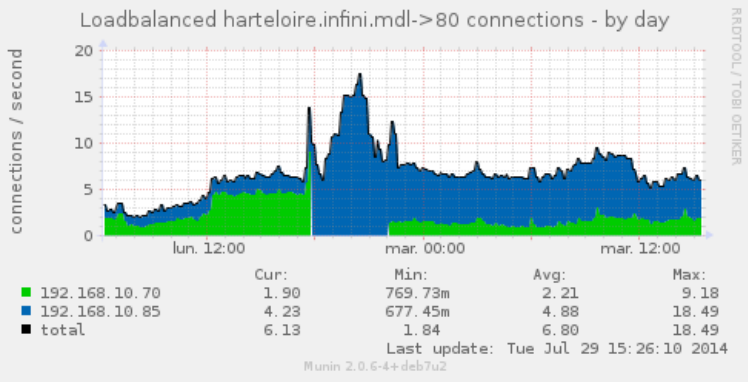
Extraire des statistiques

L'option **--list** (abréviations : -l ou -L) permet de lister toutes les adresses virtuelles portées par votre répartiteur de charge avec le nombre de connexions en cours pour chacun des serveurs qu'elles contiennent. En y ajoutant l'option **--stats**, vous aurez en plus des statistiques réseau :

```
# ipvsadm -ln
 -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
```

```
TCP  192.168.10.9:443 rr persistent 1
  -> 192.168.10.70:443            Route   1      0          0
  -> 192.168.10.85:443            Route   1      0          0
TCP  192.168.10.9:80 rr persistent 50
  -> 192.168.10.70:80             Route   1      416        136
  -> 192.168.10.85:80             Route   1      48         91
```

La plupart des outils de monitoring savent interpréter ces chiffres pour sortir des graphes qui peuvent vous être utiles, par exemple voici ce que donne un suivi de nombre de connexions en cours avec munin[W] :



Manipuler vos adresses virtuelles dynamiquement

Si vous avez décidé d'utiliser le paramètre *quorum_down* pour adapter votre architecture dynamiquement, vous pouvez vouloir ajouter des serveurs dans la liste de ceux portés par une adresse virtuelle dynamiquement. Pour cela, il faut utiliser l'option **--add-server** (abrégeable en -a), il y a bien évidemment une option **--delete-server** pour faire l'inverse :

```
# ajout du serveur 192.168.10.44
# à l'adresse virtuelle 192.168.10.9:80
 ipvsadm -a -t 192.168.10.9:80 -r 192.168.10.75

# retrait du serveur 192.168.10.44
# de l'adresse virtuelle 192.168.10.9:80
 ipvsadm -d -t 192.168.10.9:80 -r 192.168.10.75
```

---

1. Source : Article Répartition de charge[W] de Wikipédia en français - Liste des auteurs[W] ↩

(56 commentaires). Markdown Epub

 # Contournement de déficiences du Web
Posté par ¿ **Tanguy Ortolo** (**page perso**) le 23/07/14 à 14:09. Évalué à 10 (+11/-2) .

> À noter que tout cela s'applique à un service particulier, le Web, et que cette complexité n'est rendu nécessaire que par un défaut de conception du protocole HTTP et son incapacité à évoluer.
>
> Ainsi, avec d'autres services qui bénéficient protocoles plus moderne à ce regard, tels que SMTP ou XMPP, la disponibilité peut être améliorée de façon infiniment plus simple, en installant simplement un second serveur, en faisant ce qu'il faut pour qu'il collabore avec le premier, et en publiant un simple enregistrement DNS (MX ou SRV) qui indique ce serveur secondaire. Pas besoin de répartiteur de charge, et surtout pas de SPoF lié à ce répartiteur, ce qui permet de faire de la *vraie* redondance instantanée sur toute la chaîne, ce qui est à peu près infaisable pour le Web.

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **Benoît Sibaud** (**page perso**) le 23/07/14 à 14:18. Évalué à 4 (+1/-0) .

> On peut trouver la même chose sur du SIP par exemple (une vIP pour répartir sur X serveurs, test de vie sur chaque serveur, etc.). Éventuellement l'équipement matériel ou la VM fait autre chose, genre SBC, mais la fonction de répartition de charge avec test de vie est bien là.

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **Xavier Claude** (**page perso**) le 23/07/14 à 15:06. Évalué à 4 (+2/-1) .

> Tu dis ça parce que tu n'écris pas tes mails en direct, parce que ce n'est pas du tout adapté au web, si le premier MX tombe, tu va devoir attendre tout le temps du timeout avant de passer à un autre, pour le web où les gens veulent de l'immédiat, ça ne serait pas tenable.
>
> --
> « Moi, lorsque je n'ai rien à dire, je veux qu'on le sache. » Raymond Devos

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **Tanguy Ortolo** (**page perso**) le 23/07/14 à 15:22. Évalué à 1 (+1/-3) .

> Parce qu'un répartiteur de charge ne va pas attendre de timeout avant d'interroger un second serveur ?

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **Xavier Claude** (**page perso**) le 23/07/14 à 15:31. Évalué à 7 (+4/-0) .

> Si mais il peut se permettre d'avoir un timeout beaucoup plus bas (vu qu'il maîtrise son réseau derrière et que de toute façon, on ne peut pas le changer chez le client). De plus, seulement le premier client aura le temps de timeout complet, après celui-là, le serveur sera déclaré HS et tous les autres clients n'auront pas le problème.
>
> --
> « Moi, lorsque je n'ai rien à dire, je veux qu'on le sache. » Raymond Devos

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **KiKouN** le 23/07/14 à 16:31. Évalué à 4 (+2/-0) .

> Et surtout, c'est un timeout pour tout les clients et pas un timeout par client.

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **Xavier Claude** (**page perso**) le 23/07/14 à 15:36. Évalué à 4 (+1/-0) .

> Un autre point que j'ai oublié, c'est le test applicatif. Comme dit dans la dépêche, on peut tester le succès de la modification d'une page de wiki pour vérifier que le service est toujours rendu correctement. Si l'édition n'est plus possible, on vire le serveur de la liste. Comme tu fais la même chose avec l'équivalent du MX ? Le client va toujours se retrouver sur le serveur qui fonctionne à moitié sans que l'hébergeur puisse le changer (bon, il peut réattribuer l'IP mais on revient exactement aux mêmes solutions que pour le HTTP).
>
> --
> « Moi, lorsque je n'ai rien à dire, je veux qu'on le sache. » Raymond Devos

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **Tanguy Ortolo** (**page perso**) le 23/07/14 à 16:06. Évalué à 3 (+3/-3) .

> Attention, je n'ai pas dit que l'utilisation d'enregistrements de type SRV était la panacée, en revanche, cela permet de répondre de façon extrêmement simple et raisonnablement efficace à des cas d'usage très courants.
>
> Et donc, surtout, cela permet d'une vraie redondance, qui sans cela ne peut être mise en place qu'avec des délais d'indisponibilité incompressibles en poussant jusqu'à l'utilisation de routes dynamiques par BGP.

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **hermenegilde** le 23/07/14 à 16:24. Évalué à 4 (+3/-0) .

> C'est utilisé par xmpp, ldap et sip, mais je vois pas en quoi c'est la faute de http si les navigateurs n'utilisent pas SRV pour les services Web. Des tickets ont pourtant été ouverts dans Mozilla et Chromium (et Webkit).
>
> https://bugzilla.mozilla.org/show_bug.cgi?id=14328
> https://code.google.com/p/chromium/issues/detail?id=22423
>
> Que je n'ai pas lu, hein, donc je ne sais pas quel est l'avancement du sujet dans ces tickets.
>
> De plus pour ce qui est de la "simplicité", c'est bien d'avoir un système comme le SRV pour la répartition et la haute dispo, mais faut toujours bien répliquer les données derrière les services et assurer ce service de bout en bout (la "haute dispo" smtp par champ MX, ça permet tout de même pas à l'utilisateur de lire le mail quand tu utilises un MX backup non relié à ton MDA).

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **Tanguy Ortolo** (**page perso**) le 23/07/14 à 16:57. Évalué à 0 (+1/-4) .

> > Que je n'ai pas lu, hein, donc je ne sais pas quel est l'avancement du sujet dans ces tickets.
>
> Ça n'avance pas parce que l'utilisation d'enregistrements SFV avec HTTP n'a pas été normalisée. Et oui, c'est la faute de HTTP du coup.

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **hermenegilde** le 23/07/14 à 17:08. Évalué à 6 (+5/-0) .

> Non, les RFC HTTP ne spécifient pas la manière dont doit être résolu le nom du serveur. Donc l'utilisation du SRV est hors scope de la norme.

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par ☺ **Tanguy Ortolo** (**page perso**) le 23/07/14 à 17:11. Évalué à -3 (+2/-8) .

> Si, parce que l'usage d'enregistrements SRV modifie non seulement la résolution, mais également la façon d'établir les connexions, et lie d'ailleurs les deux.

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **hermenegilde** le 23/07/14 à 17:18. Évalué à 10 (+9/-0) .

> WTF. Tu as eu des cours de réseau quand tu étais jeune? Révise les.
>
> La différence entre DNS, TCP et HTTP est quand même assez bien décrite un peu partout pour que les mécanismes soient clairs, bien définis et bien séparés.
> Le navigateur lie tout ça, mais la norme HTTP ne s'occupe pas de comment est ouverte la connexion, ni comment les noms sont résolus.

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par ☺ **Tanguy Ortolo** (**page perso**) le 23/07/14 à 17:37. Évalué à 0 (+6/-9) .

> Je laisserai de côté ta moquerie, mais c'est toujours agréable, merci.
>
> Je viens de vérifier, la RFC 2616 qui définit HTTP ne mentionne pas comment les connexions sont établies, donc c'est bon en effet. Si elle indiquait ne serait ce que « le client se connecte au serveur, puis […] », ce serait fichu, parce que cela impliquerait que la résolution préalable a déterminé un unique serveur et qu'il faudrait dès lors s'y tenir.

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **CHP** le 24/07/14 à 12:02. Évalué à 5 (+3/-0) .

> Sa moquerie n'était pas agréable, mais elle a été utile : tu as vérifié et t'es rendu compte qu'il avait raison :)

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **Xavier Claude** (**page perso**) le 24/07/14 à 12:07. Évalué à 7 (+4/-0) .

> Ce n'était surtout pas la première fois que quelqu'un lui disait exactement la même chose.
>
> –
> « Moi, lorsque je n'ai rien à dire, je veux qu'on le sache. » Raymond Devos

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **django** le 23/07/14 à 17:04. Évalué à 2 (+0/-0) .

> +1 Les serveurs LDAP font de la réplication multi-maîtres [W], alors certaines de ces recettes ne leur sont pas applicables.
> Je pense que la seule qui reste valable est celle sur la répartition de charge.
>
> A noter que ce n'est pas une bonne idée de faire du round-robin sur des serveurs multi-maîtres en répétant la suite d'opérations Add-Modify-Delete sur une entrée avec le même nom. Cela ne va pas bien se passer. Oui, oui, c'est du vécu avec des clients.

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **Yves** (**page perso**) le 24/07/14 à 10:18. Évalué à 0 (+0/-0) .

> N'est-il pas possible de faire plus ou moins l'équivalent pour un serveur web, en faisant du « DNS round-robin » ?
> (À condition bien sûr que les contenus web servis soient « state-less »…)
>
> En tout cas, merci pour cet article très intéressant !

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **hermenegilde** le 24/07/14 à 11:01. Évalué à 2 (+1/-0) .

> Les navigateurs ont un cache DNS, donc pendant qques temps le navigateur ira vers le même serveur (60s sous Firefox par exemple).
>
> De plus, le serveur DNS ne sait pas si le serveur est up ou pas, tu fais de la répartition de charge, mais pas du fail over, potentiellement, tu vas impacter la moitié de tes utilisateurs pendant le temps des différents caches DNS.
>
> Alors certes, il y a des serveurs DNS qui font des tests de vie, mais pour de l'hébergement personnel, ça nécessite une infra encore plus compliquée à mettre en place. En libre je n'en connais pas, sinon cf http://www.a10networks.com/products/global-server-load-balancing.php .

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **Arcaik** le 26/07/14 à 16:17. Évalué à 1 (+1/-2) .

> > Les navigateurs ont un cache DNS, donc pendant qques temps le navigateur ira vers le même serveur (60s sous Firefox par exemple).
>
> J'espérai que ça soit faux, mais en fait non.
>
> Les gens du web sont vraiment… différents.

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **Antoine** le 27/07/14 à 18:28. Évalué à 3 (+1/-0) .

> Vu qu'un navigateur fait typiquement des tas d'accès très rapprochés à un serveur donné, ça ne paraît pas idiot. Évidemment c'est mieux de configurer un cache DNS côté client *system-wide*, mais tous les OS ne le font pas par défaut.

Répondre

[^] # Re: Contournement de déficiences du Web
Posté par **Barret Michel** le 26/07/14 à 11:18. Évalué à 4 (+1/-0) .

> Tu as finalement envoyé un mail au groupe qui discute (discutait ?) de HTTP 2 ?
>
> --
> Je suis dans ma tour d'ivoire (rien à foutre des autres, dites moi un truc pour moi), si je ne pose pas explicitement une question pour les 99%, les kévin, les Mm Michu alors c'est que je ne parle pas d'eux.

Répondre

 # feature linuxfr
Posté par **Julien CARTIGNY** (**page perso**) le 23/07/14 à 14:51. Évalué à 4 (+5/-1) .

> Ce serait bien d'avoir une feature dans linuxfr "sauvegarder cet article", un peu comme dans reddit, c'est le genre de post que je voudrais retrouver facilement.

Répondre

[^] # Re: feature linuxfr
Posté par **Tanguy Ortolo** (**page perso**) le 23/07/14 à 14:59. Évalué à 4 (+7/-6) .

> Pour sauvegarder une page, sur à peu près tous les navigateurs : Ctrl + S.
> Pour marquer une page, sous Firefox (je ne sais pas quel est le raccourci pour les autres) : Ctrl + D.

Répondre

[^] # Re: feature linuxfr
Posté par **keyser.dyson** le 24/07/14 à 18:30. Évalué à 0 (+1/-1) .

> En passant, j'ai trouvé cette commande pour exporter les favoris en ligne commande:
>
> ```
> sqlite3 ~/.mozilla/firefox/*default/places.sqlite "select a.url, a.title from moz_places a, moz_bookmarks b where a.id=b.fk and b.parent=2;"
> ```
>
> Utile si l'on souhaite éviter deux clics.

Répondre

[^] # Re: feature linuxfr
Posté par **Marotte** le 24/07/14 à 20:27. Évalué à 4 (+2/-0) .

> Oui enfin tu te rends bien compte que ça ne fait pas la même chose… Si tu veux accéder à cette page quand t'es sur le poste d'un pote/collègue/client ou dans un cyber t'auras pas les bookmarks de ton navigateur…

Répondre

[^] # Re: feature linuxfr
Posté par **Xavier Claude** (**page perso**) le 23/07/14 à 15:01. Évalué à 6 (+3/-0) .

> Il faut proposer l'entrée dans le suivi pour ça
>
> --
> « Moi, lorsque je n'ai rien à dire, je veux qu'on le sache. » Raymond Devos

Répondre

[^] # Re: feature linuxfr
Posté par **hermenegilde** le 23/07/14 à 16:13. Évalué à 1 (+1/-1) .

> Ton navigateur doit avoir une fonctionnalité de bookmark. Le raccourci ctrl-d doit même être relativement courant pour bookmarker une page.

Répondre

[^] # Re: feature linuxfr
Posté par **CHP** le 24/07/14 à 12:04. Évalué à 2 (+1/-1) .

> Les bookmarks sont locaux. Si je perds mon ordi, si je change d'ordi, etc, c'est perdu. La fonctionnalité qu'il propose permettrait de ne perdre la page en question que si on perd son compte linuxfr (et sans rien avoir à ajouter sur son poste).

Répondre

[^] # Re: feature linuxfr
Posté par **hermenegilde** le 24/07/14 à 12:12. Évalué à 2 (+2/-1) .

> Mais non! Tu peux autohéberger un serveur mozilla sync: https://wiki.archlinux.org/index.php/Mozilla_Firefox_Sync_Server
>
> Il semble même y avoir un plugin Mozilla Sync pour Owncloud: http://apps.owncloud.com/content/show.php/Mozilla+Sync?content=161793

Répondre

[^] # Re: feature linuxfr
Posté par **CHP** le 24/07/14 à 12:13. Évalué à 2 (+1/-1) .

> Quand je disais "(et sans rien avoir à ajouter sur son poste).", fallait comprendre "ni sur son poste, ni sur un serveur autre, ni sur quoi que ce soit".
>
> La feature proposée permet cela : t'installe rien nulle part, mais tu ne perds la page en question que si tu perds ton compte linuxfr. Moi je trouve que c'est une super idée.

Répondre

[^] # Re: feature linuxfr
Posté par **mael** le 24/07/14 à 16:23. Évalué à 1 (+0/-0) .

> Anéfé. Ca existe sur certains sites (je pense notamment à reddit), et c'est super pratique. En plus, je pense que ça doit être assez simple à implémenter.

Répondre

[^] # Re: feature linuxfr
Posté par **Antoine** le 27/07/14 à 18:30. Évalué à 2 (+0/-0) .

> Tu peux utiliser le service Sync par défaut, qui est hébergé par Mozilla et ne requiert pas d'installer quoi que ce soit sur quelque machine.

Répondre

 # feature linuxfr
Posté par **papatte3** le 23/07/14 à 16:02. Évalué à -10 (+4/-14) .

> Suite aux divers événements récents, est-il prévu de profiter des indications de cette dépêche pour améliorer la disponibilité du site linuxfr ?

Répondre

[^] # Re: feature linuxfr
Posté par **Anthony Jaguenaud** le 23/07/14 à 16:05. Évalué à 2 (+3/-3) .

> > Suite aux divers événements récents, est-il prévu de profiter des indications de cette dépêche pour améliorer la disponibilité du site linuxfr ?
>
> J'ai raté des indisponibilité du site ? ou tu fais de la désinformation gratuite ?

Répondre

[^] # Re: feature linuxfr
Posté par **papatte3** le 23/07/14 à 17:04. Évalué à -3 (+2/-5) .

> https://linuxfr.org/users/philippemc/journaux/this-place-has-been-accident-free-for-000-days
> https://linuxfr.org/suivi/lenteurs-et-problemes-d-acces

Répondre

[^] # Re: feature linuxfr
Posté par **Anthony Jaguenaud** le 23/07/14 à 17:17. Évalué à 5 (+4/-1) .

> > https://linuxfr.org/users/philippemc/journaux/this-place-has-been-accident-free-for-000-days
>
> Mars 2013, il y a plus d'un an… quand même, quand tu n'aimes pas, tu deviens archéologue.
>
> Qu'est-ce qui fait que tu restes ? Le syndrome de stockholme ?

Répondre

[^] # Re: feature linuxfr
Posté par **Grégoire G** (**page perso**) le 23/07/14 à 18:04. Évalué à 4 (+3/-1) .

> Il y a quelques semaines, c'était insupportables.
> Il y a eu un journal à ce propos.

Répondre

[^] # Re: feature linuxfr
Posté par **papatte3** le 24/07/14 à 09:13. Évalué à -5 (+0/-5) .

> > Qu'est-ce qui fait que tu restes ? Le syndrome de stockholme ?
>
> Simplement le fait que les gens comme toi sont très minoritaires sur le site.

Répondre

[^] # Re: feature linuxfr
Posté par **nicolas schonbrood** le 23/07/14 à 17:12. Évalué à 5 (+5/-2) .

> C'est pas toi qui voulait que les admins coupent tout le site en attendant plus d'informations ?
>
> --
> Tous les nombres premiers sont impairs, sauf un. Tous les nombres premiers sont impairs, sauf deux.

Répondre

[^] # Re: feature linuxfr
Posté par **SKy** (**page perso**) le 25/07/14 à 09:35. Évalué à 3 (+1/-0) .

> Y a de fortes chances que la 2ème machine aurait été trouée de la même façon que la 1ère. Donc avoir de la haute dispo n'aurait rien changé au problème.
>
> Le site était disponible, juste lent, ça n'a donc rien à voir avec la disponibilité, juste l'utilisabilité.
>
> --
> It's a fez. I wear a fez now. Fezes are cool !

Répondre

 # Titre de l'article
Posté par **hermenegilde** le 23/07/14 à 16:49. Évalué à 10 (+9/-0) .

> Le titre est trompeur je trouve. C'est selon moi plus un article mode d'emploi sur LVS que sur la dispo des services.
>
> - Le titre "disponibilité des services", et on parle répartition de charge. Les deux sujets sont connexes, mais on peut faire l'un ou l'autre, pas forcément les deux.
> - On ne parle que de LVS alors que les outils de répartition logicielle sont parfois plus simples à prendre en main (comme haproxy ou apache).
> - On ne parle pas des autres services (NFS, Base de données).
>
> Un article sur ces sujets est prévu Denis?

Répondre

[^] # Re: Titre de l'article
Posté par **kna** le 23/07/14 à 20:52. Évalué à 1 (+0/-0) .

> Oui, d'ailleurs, si veut faire de la « disponibilité de services », il faut 2 repartiteurs de charge en failover, pour ne pas avoir de SPOF.
>
> Idem pour les éventuels serveurs supplémentaires (NFS, BDD).

Répondre

[^] # Re: Titre de l'article
Posté par **Tanguy Ortolo** (**page perso**) le 24/07/14 à 10:11. Évalué à 3 (+0/-0) .

> > Oui, d'ailleurs, si veut faire de la « disponibilité de services », il faut 2 repartiteurs de charge en failover, pour ne pas avoir de SPOF.
>
> Pour avoir un SPoF de moins, parce que tu en auras toujours un avec ces techniques : ton FAI.

Répondre

[^] # Re: Titre de l'article
Posté par **Xavier Claude** (**page perso**) le 24/07/14 à 11:13. Évalué à 4 (+1/-0) .

> Tu peux en avoir plusieurs.
>
> --
> « Moi, lorsque je n'ai rien à dire, je veux qu'on le sache. » Raymond Devos

Répondre

[^] # Re: Titre de l'article
Posté par **Tanguy Ortolo** (**page perso**) le 24/07/14 à 11:24. Évalué à 3 (+1/-1) .

> Avec des routes dynamiques par BGP donc. Ça implique d'avoir son AS, et la propagation de nouvelles routes n'a rien d'instantané.

Répondre

[^] # Re: Titre de l'article
Posté par **Denis Dordoigne** le 23/07/14 à 21:17. Évalué à 3 (+4/-3) .

> Le titre est trompeur je trouve. C'est selon moi plus un article mode d'emploi sur LVS que sur la dispo des services.

Je suis développeur de formation, donc j'écris les présentations techniques comme je les ai toujours vues : d'abord une présentation des concepts, ensuite l'explication du fonctionnement et enfin un exemple d'implémentation (comme à l'école j'ai appris les algorithmes de tri d'abord en intégrant les concepts de tableau et d'arbres, ensuite en apprenant les différentes méthodes de tri puis enfin leur implémentation en C : pour moi j'ai suivi un cours d'algorithmique, pas de C) ; je sais que ça peut surprendre les administrateurs système qui sont habitués aux formations professionnelles purement techniques, pour ma part j'en ai suivi beaucoup et malheureusement je n'en ai pas retenu grand chose : j'écris pour les gens qui ont le cerveau aussi mal foutu que le mien :)

> Le titre "disponibilité des services", et on parle répartition de charge

Je ne comprends pas cette remarque : à quoi peut bien servir la répartition de charge si ce n'est pas pour améliorer la disponibilité de ses services ? Parmi mes nombreux défauts j'ai aussi celui de ne pas être un geek, je ne fais de la technique que pour répondre à un besoin, pas pour la beauté conceptuelle ou technique du geste.

> On ne parle que de LVS alors que les outils de répartition logicielle sont parfois plus simples à prendre en main (comme haproxy ou apache).
> On ne parle pas des autres services (NFS, Base de données).
> Un article sur ces sujets est prévu Denis?

Il fallait faire des choix, l'article est déjà très long, mais pour information une dépêche concernant haproxy est dans l'espace de rédaction, si le sujet t'intéresse tu peux y contribuer. Concernant les bases de données, j'attends la sortie de mariadb 10.1 (qui devrait intégrer galera et enfin fonctionner correctement en mode multi-maître) mais a priori mes prochaines grosses dépêches concerneront inotify et la mise en place d'un firewall.

--
En slip dans http://www.youtube.com/watch?v=aHJYsfdsCLg

Répondre

[^] # Re: Titre de l'article
Posté par **hermenegilde** le 23/07/14 à 21:57. Évalué à 7 (+6/-0) .

> > Le titre "disponibilité des services", et on parle répartition de charge
>
> Je ne comprends pas cette remarque : à quoi peut bien servir la répartition de charge si ce n'est pas pour améliorer la disponibilité de ses services ? Parmi mes nombreux défauts j'ai aussi celui de ne pas être un geek, je ne fais de la technique que pour répondre à un besoin, pas pour la beauté conceptuelle ou technique du geste.

Ce n'est pas du tout conceptuel, ce sont deux choses totalement différentes, la répartition de charge et la gestion de la haute dispo. Connectées, mais différentes.

Côté répartition de charge, tu l'as expliqué, on a le round robbin, le least connection, bref un ensemble de techniques pour distribuer la charge plus ou moins justement sur un ensemble de serveurs, tous actifs.

Côté haute dispo, la virtual ip du service qui se balade du serveur actif au passif, la bascule d'une BDD avec linux ha (ou l'équivalent à la mode du moment), la redondance du stockage.

On fait plus souvent de la haute dispo sans répartition de charge, pour s'assurer la disponibilité d'une base de donnée par exemple. On met un mysql ou un postgresql en cluster actif/passif, et on bascule les clients (par modif de conf et arrêt/relance, ou par gestion de la couche client qui se connecter).

La répartition de charge sans haute dispo, google fait ça. Il faut du shared nothing sur des zilliards de serveur, ça monte très bien en charge, on peut rajouter des noeuds sans s'embêter. Et tant pis si le crash d'un serveur prive de service 0.1% des utilisateurs. Parce que c'est compliqué de faire de la haute dispo sur toute la chaine, c'est souvent plus simple, moins complexe à gérer, d'accepter une certaine perte de service pour une partie des utilisateurs.

Là dans la dépêche par exemple, on parle surtout faire du Load balancing / high availibility sur du service HTTP. Mais ça parle du serveur NFS pour stocker les sessions utilisateurs. Si le serveur NFS n'est pas là, tout le service est indisponible (en tout cas, le test de vie applicatif devrait échouer). Le maillon le plus faible conditionne la disponibilité de l'ensemble.

La répartition de charge sur des services autre que http, c'est moins courant par contre. Côté BDD relationnelle on a tendance à pas trop s'embêter avec ça, et si on veut pas de relationnel (ou qu'on peut s'en passer) il reste les quelques nosql prévus pour ça, comme Cassandra ou Couchdb.
Côté système de fichier, les rares qui gèrent ça (de ma faible expérience, j'en ai pas testé des masses) ont tendance à être compliqués à mettre en place, pour des perfs assez aléatoires.

On peut toujours mettre un test de vie et envoyer les utilisateurs vers un autre serveur, mais comme les serveurs ne partagent rien, l'utilisateur perd la session, et donc ce qu'il était en train de faire. La haute dispo du pauvre.

Répondre

 # Pour la persistence de session
Posté par **reno** le 23/07/14 à 16:50. Évalué à 2 (+0/-0) .

> Je crois qu'il y a des algorithmes plus intelligent que mémoriser les serveurs pour tout les clients, par exemple commencer par faire un algorithme qui fait un hash des informations du client et qui s'il tombe sur un serveur indisponible (planté ou plein) va appliquer une deuxième étape.
>
> Il y a aussi le problème lié a l'augmentation du nombre de serveur a gérer si possible de manière transparente.

Répondre

 # Pertinence de l'IP aliasing sur l'interface de bouclage ?
Posté par **Bernez** le 24/07/14 à 14:00. Évalué à 2 (+1/-0) .

> > Ensuite, il convient de faire comprendre au serveur qu'il gère l'adresse IP virtuelle. Le plus propre pour faire cela est de déclarer celle-ci comme un alias de l'interface de loopback
>
> En quoi faire ça est plus propre que de faire de l'IP aliasing sur l'interface réseau qui sert effectivement à connecter le serveur au répartiteur de charge ? Ça me paraît plus sale au contraire.

Répondre

[^] # Re: Pertinence de l'IP aliasing sur l'interface de bouclage ?
Posté par **KiKouN** le 24/07/14 à 14:38. Évalué à 3 (+1/-0) .

> Si tu mets l'IP (du service) sur l'interface connecté au réseau. Tu va émettre et répondre aux requêtes ARP sur ce réseau et piquer réellement l'IP au répartiteur de charge.
> Bon tu me dira, le serveur à côté va faire pareil. Il y aura alors une répartition de charge au bon vouloir des serveurs mais les doublons d'IP c'est très chiant.
>
> Le plus propre pour moi est de faire une réécriture d'adresse (IP du service -> IP serveur) au niveau des serveurs.

Répondre

[^] # Re: Pertinence de l'IP aliasing sur l'interface de bouclage ?
Posté par **Bernez** le 24/07/14 à 16:44. Évalué à 3 (+2/-0) .

> > Si tu mets l'IP (du service) sur l'interface connecté au réseau. Tu va émettre et répondre aux requêtes ARP sur ce réseau et piquer réellement l'IP au répartiteur de charge.
>
> En effet. Alors on peut utiliser une interface "dummy". Je trouverais ça moins bizarre que squatter "lo".

Répondre

[^] # Re: Pertinence de l'IP aliasing sur l'interface de bouclage ?
Posté par **Denis Dordoigne** le 24/07/14 à 16:58. Évalué à 3 (+1/-0) .

> L'usage d'un alias de lo s'est imposé du temps de linux 2.0 et 2.2 avec lesquels les interfaces dummy ne savaient pas communiquer en ARP. Certes ce n'est pas forcément aujourd'hui le plus pertinent, mais comme c'est un usage les habitués des répartiteurs de charge avec routage direct ne sont pas choqués en voyant ça, voire au contraire comprennent immédiatement ce dont il s'agit.
>
> –
> En slip dans http://www.youtube.com/watch?v=aHJYsfdsCLg

Répondre

[^] # Re: Pertinence de l'IP aliasing sur l'interface de bouclage ?
Posté par **madgeekone** le 25/07/14 à 14:08. Évalué à 3 (+3/-0) .

> Salut à tous
> Bel article Messieurs. Et en plus on voit que ça intéresse du monde.
>
> Effectivement, pour avoir pratiquer un peu le load balancing et la haute dispo en général je ne trouve pas du tout choquant d'utiliser la lo.
> Voir même je trouve ça ingénieux.
> L'autre intérêt est que, si l'on s'y prend bien, l'adresse IP attachée à la lo sera UP en permanence, même si on perd eth0 en reroutant le trafic par eth1.
> D'ailleurs dans le monde du réseau, il est très (mais alors très) fréquent d'utiliser des loopback sur de routeurs. (routing, multicast, voix,etc)
> Et dans notre cas, notre serveur Linux devient un équipement réseau..Un load balancer est plus souvent une applicance qu'un soft.
>
> Je ne vais surement rien vous apprendre, mais pour ceux que ça inquiète, l'utilisation d'un load balancer n'est pas forcément synonyme de SPOF.
> Avec des outils comme keepalived on peut facilement redonder les VIP en utilisant vrrp.
>
> Et avec un couple d'outil comme pound et keepalived, vous avez de quoi vous faire un lb hautement disponible. Et on est même pas obligé d'utiliser la lo pour ceux qui ca choquent :D
>
> ./MadGeeK:~#

Répondre

VM
Posté par **keyser.dyson** le 24/07/14 à 18:49. Évalué à 1 (+2/-1) .

> On peut aussi injecter les requêtes dans un réseau virtuel et mapper les architectures souhaitées dans les machines virtuelles.

Répondre

Et pas que HTTP.....
Posté par **madgeekone** le 25/07/14 à 14:12. Évalué à 0 (+1/-1) .

> Resalut
> Pour info, même si on parle là de HTTP, la quasi totalité des protocoles de TCP et UDP sont load balancable (oui je sais c'est pas très beau à lire..ni à écrire d'ailleurs)
>
> Pour peu qu'on sache prendre en charge les sticky de toute sorte (dns, tcp, cookie, etc.) on peut imagnier load-balancer "presque" tout ce qu'on veut.
>
> ./MadGeek:~#

Répondre

Envoyer un commentaire | Suivre le flux des commentaires

Note : les commentaires appartiennent à ceux qui les ont postés. Nous n'en sommes pas responsables.

Revenir en haut de page

À propos de LinuxFr.org

- Mentions légales
- Faire un don
- Team LinuxFr.org
- Informations sur le site
- Aide / Foire aux questions
- Règles de modération
- Statistiques
- API pour les développeurs
- Code source du site
- Plan du site